REMARKS

This is a response to the non-final Office Action mailed June 26, 2008.

Claims 1, 8, 10, 13, 17, 20, 24, 27, 32, 36, 41, 45, 47, 51, 69, 70, 76 and 81 are amended, and claim 83 is new. No new matter is presented. Example support for the claims is as follows: claims 1 and 41 (p.7, lines 18 and 19; p.13, lines 4 and 5), claims 8 and 51 (claim 28), claims 10, 20, 27, 32, 36, 47 and 76 (p.25, lines 4-12), claim 13 (p.26, lines 24-27), claims 17 and 24 (p.3, lines 25-27), claim 69 (p.28, lines 13-25) and claim 76 (p.24, lines 11-14; p.25, lines 4-12).

Paragraph 5 of the Office Action

Claim 81 has been rejected under 35 U.S.C. §112 as being indefinite for failing to point out and distinctly claim the subject matter which applicant regards as the invention. Claim 81 is amended in response to the rejection. Withdrawal of the rejection is therefore respectfully requested.

Paragraph 9 of the Office Action

Claims 1, 5, 6, 8-10, 13, 21-24, 27, 28, 30-32, 37-40, 45-51, 53, 54, 67, 69, 70, 76 and 81 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Tuli (US 7,068,381) in view of Davis (US 6,643,696). Applicants respectfully traverse this and the other rejections.

Tuli provides a portable Internet access device which allows a user to view a bitmap image of a web page. Tuli's goal is to avoid the need for a powerful microprocessor which provides a mini-browser in the device which interprets and displays information received from the Internet. Instead, Tuli's device allows a user to request to view a web page, where the request is handled by a web server program 2 at a host 1. The web server program receives HTML, JAVA etc. information and provides it to a browser translator 4, which translates the information (i.e., the entire image comprising graphics and text) to a black and white bit map or raster image. The image is divided after the bit map is created. The image is then sent to a compression program 11 at the host which compresses the image. The compressed information is finally provided to the user device.

A CPU in the user device has the ability to decompress the bit map or raster image and display an image. The user may traverse portions of the image on a display of the user device by

scrolling. The CPU does not have the ability to identify links on the displayed image. However, the words that represent links can be displayed in bold, and a user selection via a pointing device can be recognized. The user selection is transmitted to a program 14 at the host 1 to obtain a new web page. See, e.g., col. 1, line 66 to col. 3, line 19, and Fig. 1.

In contrast, Applicants' claim 1 sets forth accessing a mark-up language description of particular content, where the mark-up language description includes one or more source files which describe a behavior of the particular content on a user interface of a user device based on user interactions with the particular content via the user interface. In contrast, the HTML, JAVA etc. information which is translated to bit map or raster image by Tuli does not describe a behavior of the particular content on a user interface of a user device based on user interactions with the particular content via the user interface. Instead, the bit map or raster image is simply image data and does not describe any behavior as claimed.

Further, claim 1 sets forth compiling a mark-up language description of particular content…to create executable code for a user device. In contrast, *the translating process which is performed by Tuli is not compiling, and the bit map or raster image data is not executable code*, contrary to the assertion in the Office Action, p.4, 1st full paragraph. The definitions of some of the claims terms set forth in the Office Action are not consistent with the plain meaning of the claims. See MPEP 2111.01. For example, executable code can include byte code or object code (see, e.g., Applicants' specification, p.3, lines 7 and 8). Further, an "executable program" is defined as "a program that can be run" in the Microsoft Computer Dictionary (2002), see attachment. Also, "code" is defined as "program instructions" (see attachment).

Further, claim 1 sets forth transmitting executable code to a user device for execution by the user device to provide particular content via a user interface according to one or more source files, and according to a behavior which is described by the one or more source files and according to user interactions. Again, Tuli simply displays an image and does not provide content according to a described interactive behavior as claimed. In contrast, Applicants' approach can provide feature rich dynamic interactions (specification, p.2, lines 27 and 28).

Davis provides a process for tracking client interaction with a resource such as a web page downloaded from a server. In particular, an HTML document is downloaded from a server (e.g., server A, Fig. 3), then images specified by first embedded URLs are downloaded. A second

embedded URL in the document points to a first executable program that runs on a server (e.g., server B). A third embedded URL in the document points to a second executable program that is downloaded to and runs on the client. When the images specified by the first embedded URLs are downloaded, the first executable program on the server runs. The server can capture identifying information from the client. The second executable program determines the current time when it initializes, and the current time when the user leaves the HTML document. The elapsed time is then reported to the server (col. 5, line 40-col. 6, line 16). Davis is not concerned with describing the behavior of content on a user interface based on user interactions with the content via the user interface. Davis only indicates that his tracking scheme may display an extremely small and completely transparent GIF image type (col. 12, lines 24-27).

Regarding the suggestion that it would be obvious to combine Tuli and Davis, such an attempt would fail since Tuli's user device does not have the ability to execute a downloaded program. Further, regarding the asserted motivation to combine Tuli and David to provide more complex content on a thin client device, Tuli specifically is concerned with a low cost device which does not need a powerful microprocessor and therefore seeks to avoid additional complexity, as mentioned previously.

Claim 1 and its dependent claims are therefore clearly patentable.

Claim 5 sets forth that compiling includes converting data in a markup language document to ActionScript and compiling the ActionScript into ActionScript byte code. Applicants respectfully submit that the use of Official Notice is improper. The notice of facts beyond the record which may be taken by the examiner must be "capable of such instant and unquestionable demonstration as to defy dispute." MPEP 2144.03. In fact, it would be not be obvious to use Tuli's browser translator to provide a compilation into ActionScript because Tuli only translates HTML, JAVA etc. information to a bit map or raster image. A bit map or raster image is not a script such as ActionScript. Further, Tuli is concerned with providing a low cost device which does not use a complex microprocessor, and therefore cannot process a script.

Moreover, the Office Action apparently interprets claim 5 as referring to compiling web data into ActionScript. In fact, claim 5 sets forth both: (1) converting data in a markup language document to ActionScript, and (2) compiling the ActionScript into ActionScript byte code. The Office Action must address each of the claimed features, including the converting feature.

Claim 5 is therefore clearly patentable.

Claim 8 sets forth accessing media content comprising at least one of audio, video and a movie, providing a reference in a mark-up language description to a media file which contains the media content, and transmitting the media file with executable code to a user device for use by a rendering entity at the user device in rendering the media content on a user interface when the media file is referenced when the executable code is executed. The Office Action asserts that Tuli can retrieve HTML information, and that Davis provides a reference to a media file which contains media content. Actually, Davis embeds an executable program in an HTML document by using a URL which points to the executable program (col. 5, lines 54-58). This executable program is a *tracking program* which determines how long a user interacts with an HTML document (col. 6, lines 5-16), and not a media file. Further, regarding the processing of data at a user device of Tuli, again, the user device of Tuli cannot render media content comprising at least one of *audio, video and a movie* since it is a low cost device which can only display a static image.

Claim 8 is therefore clearly patentable.

Claim 9 is patentable at least for the reasons set forth in connection with claim 5.

Claim 10 sets forth transcoding a media file to an accepted format before transmitting it, where the transcoding is separate from the compiling as set forth in claim 1. Tuli at col. 2, lines 25-29 refers to dividing an image into sections after a bitmap or raster of the image is created. However, this division is not *transcoding*, as Tuli performs no such transcoding.

Claim 10 is therefore clearly patentable.

Claim 13 sets forth that a first application runs on a user device, and a request is received at a server from the user device for second content when the first application which runs on the user device calls the second application. The method further includes accessing and compiling a mark-up language description of the second content, and transmitting the compiled mark-up language description to the user device. Regarding Tuli at col. 2, lines 5-13, this refers to a web server receiving a web page, electronic message, HTML or JAVA information. However, such information is not an application. Moreover, Davis at col. 5, lines 54-58 refers to a second executable program that is downloaded to, and runs on, a client. However, it does not call another application as claimed.

Claim 13 is therefore clearly patentable.

Independent claim 21 and its dependent claims, including claims 22-24 and 27, are similarly patentable based on the above comments. The Examiner is requested to clarify the rejection of claim 1 as the Office Action (p.8) refers to a second application called by a first application, but this language is not in claim 21.

Independent claim 37 and its dependent claims, including claims 38-40, are similarly patentable based on the above comments.

The Examiner is requested to clarify the rejection of claim 45 as the Office Action (pp.11 and 12) refers to a mark up language description which is not in the claim, and does not refer to specific features which are in the claim.

Independent claim 45, and its dependent claims, including claims 46 and 47, are similarly patentable based on the above comments.

The Examiner is requested to clarify the rejection of claim 48 as the Office Action (pp.13 and 14) refers to a second application which is called, which is not in the claim, and does not refer to specific features which are in the claim, such as a script code.

Independent claim 48, and its dependent claims, including claims 49 and 50 are similarly patentable based on the above comments.

Regarding claim 51, as mentioned, Tuli does not have the ability to process media data comprising at least one of audio, video and a movie.

Claim 51 is therefore clearly patentable.

Regarding claims 53 and 54, as mentioned, the image data of Tuli is not executable code. For example, claims 54 specifies that the executable code of claim 1 comprises at least one of object code and byte code. Tuli provide no mention of such executable code. Regarding Tuli at col. 4, lines 18-21, this refers to the image which is transferred to the user device from the host being a compressed color image such as a JPEG image. However, *a JPEG image does not comprise object code or byte code. JPEG is simply an image file format.*

Claims 53 and 54 are therefore clearly patentable.

Claim 67 is patentable at least for the reasons discussed previously.

Claim 69 sets forth that the compiling of claim 1 comprises parsing a markup language description to identify first and second types of elements in the markup language description. The first type of element is provided to a first compiling module which is appropriate for the first type of

element to obtain first object code, and the second first type of element is provided to a second compiling module which is appropriate for the second type of element to obtain second object code. The first and second object code is assembled into a single executable. Tuli at col. 2, lines 9-13 refers to translating information such as HTML and JAVA information to a bit map or raster image. However, there is no mention of providing first and second compiling modules, respectively, which are appropriate for first and second types of elements. Instead, Tuli translates all information, including "the entire image comprising graphics and text" received in the form of HTML, JAVA, etc. to a bit map or raster image using a translator program 4 (col. 2, lines 12-17). There is simply no disclosure or suggestion of using different compiling modules which are appropriate for different types of elements. Further, the translating is not compiling as discussed previously.

Claim 69 is therefore clearly patentable.

Claim 70 sets forth that the first type of element (in claim 69) provides a script which defines a behavior of particular content, and the second type of element defines a connection to an external data source. Regarding the suggestion to modify Tuli in view of Davis, Tuli is concerned with providing a low cost user device which avoids the need for a browser capability (col. 1, line 66 to col. 2, line 4), and which only displays image data. Accordingly, such a device cannot process a script. Also, Tuli provides the server/host 1 as a virtual browser which can access information from different sources (col. 2, lines 12-21), so that the low cost user device does not need to access external data sources. Accordingly, Tuli teaches against the proposed modification. Further, Davis provides no mention of a script.

Claim 70 is therefore clearly patentable.

Claim 76 sets forth that the transcoding of claim 27 comprises at least one of: (a) transcoding one audio format to another audio format, and (b) transcoding one video format to another video format. Tuli at col. 4, lines 18-21 mentions that the image data provided to the user device may be provided using JPEG compression. However, this does not involve audio or video compression. In fact, the user device of Tuli cannot even process audio or video data.

Claim 76 is therefore clearly patentable.

Claim 81 sets forth providing an object in the executable code which identifies at least one of a name and a format of media content, where the at least one of name and a format is provided via the user interface when said media content is rendered. While Tuli at col. 4, lines 18-21 mentions

that the image data provided to the user device may be provided using JPEG compression, there is no disclosure or suggestion of identifying the name and a format of the media content.

Claim 81 is therefore clearly patentable.

Paragraph 40 of the Office Action

Regarding paragraph 40 of the Office Action, claims 4, 7, 28, 30-33, 36, 41-44, 52, 55-58, 60-62, 64, 65, 73, 77, 78, 80 and 82 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli and Davis in view of Rubin (US 6,701,522). Rubin is cited for showing use of a plug-in to a web browser (col. 7, lines 17-23). However, it would not be obvious to provide the user device of Tuli with a plug-in to a web browser since Tuli seeks specifically seeks to provide a low cost device which *does not have a browser capability* since this requires a powerful microprocessor. Instead, Tuli uses a host 1 as a virtual browser for the user device (col. 1, line 66 to col. 2, line 4; col. 2, lines 12-21).

Withdrawal of the rejection is therefore respectfully requested.

Paragraph 68 of the Office Action

Regarding paragraph 68 of the Office Action, claim 11 has been rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli and Davis in view of Russell (US 2002/0069420). Claim 11 is allowable at least by virtue of its dependence on claim 1, which is allowed for the reasons discussed previously.

Withdrawal of the rejection is therefore respectfully requested.

Paragraph 70 of the Office Action

Regarding paragraph 70 of the Office Action, claims 14-17, 19, 20, 74 and 75 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli in view of Wagner (US 6,085,224). Wagner is cited for showing the combination of a markup language code and a scripting language description. However, even if this is true, there is still no disclosure or suggestion of compiling a markup language description and a scripting language description to create combined executable code. As mentioned previously, the translation to image data which is performed by Tuli is not compiling and the resulting image data is not executable code. Further, the user device of Tuli

is not powerful enough to process either a markup language description or a scripting language description.

Withdrawal of the rejection is therefore respectfully requested.

Paragraph 79 of the Office Action

Regarding paragraph 79 of the Office Action, claims 68 and 79 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli and Rubin in view of Davis. These claims are allowable for the reasons discussed previously.

Withdrawal of the rejection is therefore respectfully requested.

New claim

Claim 82 sets forth that the executable code of claim 1 provides a script which is executed when a specified event occurs when a user interacts with particular content via a user interface, and the specified event is based on user control of a pointing device or a key press. As mentioned, Tuli's user device does not have the ability to process scripts, and seeks to avoid the need for a powerful microprocessor. Further, while Tuli's user device allows a user to enter a command via a pointing device, such a command does not result in executing a script based on executable code at the user device. Instead, a user click on the display of the user device is sent to the host computer 1 for processing (col. 2, line 64 to col. 3, line 7).

Claim 82 is therefore clearly patentable.

Conclusion

In view of the above, each of the pending claims is believed to be in condition for immediate allowance. The Examiner is therefore respectfully requested to pass this application on to an early issue.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: September 25, 2008

By: /Ralph F. Hoppin/
Ralph F. Hoppin
Reg. No. 38,494

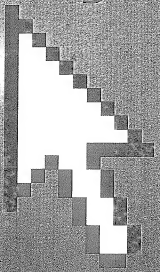
VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105-4206
Telephone: (415) 369-9660, x214
Facsimile: (415) 369-9665

Microsoft

OVER
10,000
ENTRIES

Microsoft Computer Dictionary

Fifth Edition



- *Fully updated with the latest technologies, terms, and acronyms*
- *Easy to read, expertly illustrated*
- *Definitive coverage of hardware, software, the Internet, and more!*

C

COBOL *n.* Acronym for Common Business-Oriented Language. A verbose, English-like compiled programming language developed between 1959 and 1961 and still in widespread use today, especially in business applications typically run on mainframes. A COBOL program consists of an Identification Division, which specifies the name of the program and contains any other documentation the programmer wants to add; an Environment Division, which specifies the computers being used and the files used in the program for input and output; a Data Division, which describes the format of the data structures used in the program; and a Procedure Division, which contains the procedures that dictate the actions of the program. *See also* compiled language.

cobweb site *n.* A Web site that is far out of date. *See also* Web site.

Cocoa *n.* A set of object-oriented development tools and interfaces available on Mac OS X. Cocoa contains a set of frameworks, software components, and development tools used to construct applications for Mac OS X and provides programming interfaces in Java and Objective-C. Cocoa is based on NeXT's OpenStep and is integrated with Apple technologies.

CODASYL *n.* Acronym for Conference on Data Systems Languages. An organization founded by the U.S. Department of Defense. CODASYL is dedicated to the development of data-management systems and languages, among them the widely used COBOL.

code[1] *n.* **1.** Program instructions. Source code consists of human-readable statements written by a programmer in a programming language. Machine code consists of numerical instructions that the computer can recognize and execute and that were converted from source code. *See also* data, program. **2.** A system of symbols used to convert information from one form to another. A code for converting information in order to conceal it is often called a *cipher*. **3.** One of a set of symbols used to represent information.

code[2] *vb.* To write program instructions in a programming language. *See also* program.

codo access security *n.* A mechanism provided by the runtime whereby managed code is granted permissions by security policy and these permissions are enforced, limiting what operations the code will be allowed to perform. To prevent unintended code paths from exposing a security vulnerability, all callers on the call stack must be granted the necessary permissions (possibly subject to override by assertion or denial).

codec *n.* **1.** Short for coder/decoder. Hardware or software that can compress and uncompress audio or video data. *See also* compress[2], uncompress. **2.** Hardware that combines the functions of definitions 1 and 2.

code conversion *n.* **1.** The process of translating the instructions from one form into another. Code may be converted at the source-language level (for example, from Pascal), at the hardware-platform level (for example, working on the IBM PC to working on the Apple Macintosh), or at the language level (for example, from source code in C to machine code). *See also* code[1] (definition 1). **2.** The process of transforming data from one representation to another, such as from ASCII to EBCDIC or from two's complement to binary-coded decimal.

Code Division Multiple Access *n.* A form of multiplexing in which the transmitter encodes the signal, using a pseudo-random sequence that the receiver also knows and can use to decode the received signal. Each different random sequence corresponds to a different communication channel. Motorola uses Code Division Multiple Access for digital cellular phones. *Acronym:* CDMA. *Also called* spread spectrum. *See also* multiplexing, transmitter.

code page *n.* In MS-DOS versions 3.3 and later, a table that relates the binary character codes used by a program to keys on the keyboard or to the appearance of characters on the display. Code pages are a means of providing support for character sets and keyboard layouts used in different countries. Devices such as the display and the keyboard can be configured to use a specific code page and to switch from one code page (such as United States) to another (such as Portugal) at the user's request.

code profiler *n.* A tool designed to aid developers in identifying and eliminating the code inefficiencies that cause bottlenecks and degrade performance in their applications. Code profilers analyze an executing application to determine both how long functions take to execute and how often they are called. Using a code profiler is a repetitive process in that the tool must be reused after each section of inefficient code has been found and corrected.

coder *n.* *See* programmer.

Code Red worm *n.* A fast-spreading and pernicious Internet worm first discovered in mid-2001. The Code Red

nications. The addition of SBC's Internet customer base made Prodigy the third largest ISP in the United States.

Prodigy Information Service *n.* An online information service founded by IBM and Sears. Like its competitors America Online and CompuServe, Prodigy offers access to databases and file libraries, online chat, special interest groups, e-mail, and Internet connectivity. *Also called:* Prodigy.

product *n.* **1.** An operator in the relational algebra used in database management that, when applied to two existing relations (tables), results in the creation of a new table containing all possible ordered concatenations (combinations) of tuples (rows) from the first relation with tuples from the second. The number of rows in the resulting relation is the product of the number of rows in the two source relations. *Also called:* Cartesian product. *Compare* inner join. **2.** In mathematics, the result of multiplying two or more numbers. **3.** In the most general sense, an entity conceived and developed for the purpose of competing in a commercial market. Although computers are products, the term is more commonly applied to software, peripherals, and accessories in the computing arena.

production system *n.* In expert systems, an approach to problem solving based on an "IF this, THEN that" approach that uses a set of rules, a database of information, and a "rule interpreter" to match premises with facts and form a conclusion. Production systems are also known as rule-based systems or inference systems. *See also* expert system.

P

Professional Graphics Adapter *n.* A video adapter introduced by IBM, primarily for CAD applications. The Professional Graphics Adapter is capable of displaying 256 colors, with a horizontal resolution of 640 pixels and a vertical resolution of 480 pixels. *Acronym:* PGA.

Professional Graphics Display *n.* An analog display introduced by IBM, intended for use with their Professional Graphics Adapter. *See also* Professional Graphics Adapter.

profile[1] *n. See* user profile.

profile[2] *vb.* To analyze a program to determine how much time is spent in different parts of the program during execution.

profiler *n.* A diagnostic tool for analyzing the run-time behavior of programs.

Profiles for Open Systems Internetworking Technology *n. See* POSIT.

program[1] *n.* A sequence of instructions that can be executed by a computer. The term can refer to the original source code or to the executable (machine language) version. *Also called:* software. *See also* program creation, routine, statement.

program[2] *vb.* To create a computer program, a set of instructions that a computer or other device executes to perform a series of actions or a particular type of work.

program button *n.* On a handheld device, a navigation control that is pressed to launch an application. *Also called:* application button.

program card *n. See* PC Card, ROM card.

program cartridge *n. See* ROM cartridge.

program comprehension tool *n.* A software engineering tool that facilitates the process of understanding the structure and/or functionality of computer applications. *Acronym:* PCT. *Also called:* software exploration tool.

program counter *n.* A register (small, high-speed memory circuit within a microprocessor) that contains the address (location) of the instruction to be executed next in the program sequence.

program creation *n.* The process of producing an executable file. Traditionally, program creation comprises three steps: (1) compiling the high-level source code into assembly language source code; (2) assembling the assembly language source code into machine-code object files; and (3) linking the machine-code object files with various data files, run-time files, and library files into an executable file. Some compilers go directly from high-level source to machine-code object, and some integrated development environments compress all three steps into a single command. *See also* assembler, compiler (definition 2), linker, program.

program encapsulation *n.* A method of dealing with programs with Year 2000 problems that entailed modifying the data with which a program worked. The input data is modified to reflect a parallel date in the past that the program can handle. When output is generated, that data is changed again, to reflect the correct date. The program itself remains unchanged.

program file *n.* A disk file that contains the executable portions of a computer program. Depending on its size and